PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 13/00 — A1

(11) International Publication Number: WO 99/17210

(43) International Publication Date: 8 April 1999 (08.04.99)

(21) International Application Number: PCT/US98/19611

(22) International Filing Date: 21 September 1998 (21.09.98)

(30) Priority Data:
08/938,252 26 September 1997 (26.09.97) US

(71)(72) Applicants and Inventors: TOUGHEY, Daniel, J. [US/US]; 15409 W. 93rd Terrace, Lenexa, KS 66219 (US). VERMEIRE, Dean, R. [US/US]; 11304 W. 77th Street, Shawnee, KS 66214 (US). STOCKMYER, Mark, L. [US/US]; 2611 N.E. 69th Terrace, Gladstone, MO 64119 (US). WELLAND, Scott, P. [US/US]; 12313 S. Darnell Court, Olathe, KS 66062 (US).

(74) Agent: STITT, Richard, P.; Spencer Fane Britt & Browne LLP, Suite 1400, 1000 Walnut Street, Kansas City, MO 64106 (US).

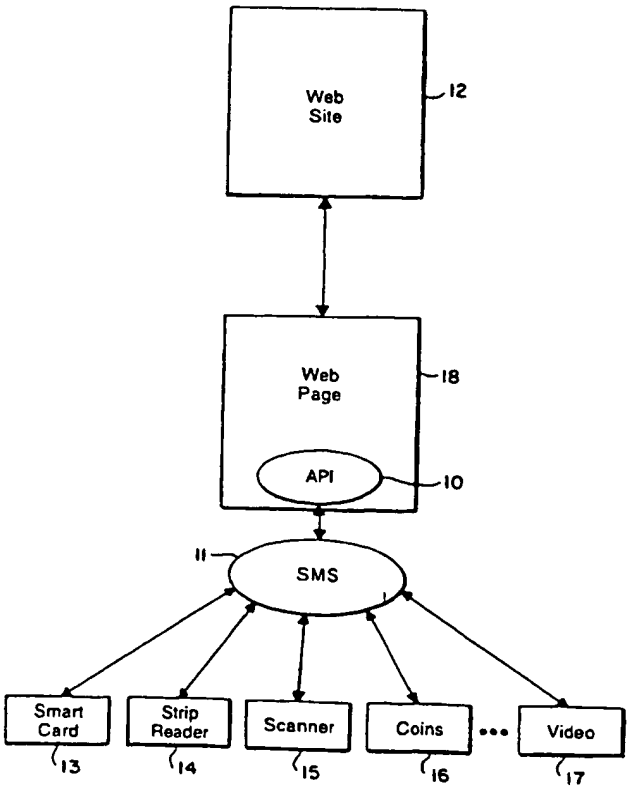
(81) Designated States: AU, CA, JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: DETECTION AND CONTROL OF NON-SERVER-BASED COMPUTER PROGRAMS

(57) Abstract

A system and method are provided for enabling a Web site (12) communicate with an optionally activate and/or control hardware and software located on an access terminal (71) without the need for a particular Web browser. The method and system consist of an applications program interface ("API") (10) which acts as a go between, translator, and/or message carrier between a standard Internet browser and local system-level code. Utilizing one of a number of protocols for Web-based functionality such as JAVA or Active-X, the API is triggered by establishment of the interconnection between the local terminal and the Web-based code, regardless of where the code resides such as on are remote Web site over the Internet, within an organization's Intranet, or on the terminal itself.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

DETECTION AND CONTROL OF NON-SERVER-BASED COMPUTER PROGRAMS

BACKGROUND OF THE INVENTION

A. Field of the Invention

The present invention relates to a browser independent system and method for detecting and controlling, from a web site, computer programs and peripherals residing, not on the Web server, but at the user's own access node or at another access node remote from the Web site.

B. Description of the Prior Art

In the past half decade the demand for access to information in this country and in the world has skyrocketed. Indeed, sociologists and anthropologists have labeled this period the dawn of the information age. As a result, means for storing, accessing, evaluating, and utilizing information have developed at an astonishing rate.

One type of information storage and retrieval system that was implemented in the 1970's and is still in use today is the proprietary database. Under this approach, an information vendor compiles data and information or licenses rights to data and information from third parties and then makes such data and information available to consumers thereof through subscription agreements. In some cases, access to such information is restricted to those subscribers who access such information through the information provider's proprietary data access terminals. Examples of such approaches to data access familiar to the legal industry include Lexis® and Westlaw.® (All trademarks identified herein are owned by third parties and no claim of ownership or association is intended by their use herein.) There are, however, many other examples (including Dialog Information Systems®), and with the personal computer revolution, most of these information providers have made access available through standard personal computers.

In addition to offering access to stored data and information and the means for retrieving and storing such information, companies offering this information sometimes also offer ancillary services such as electronic mail ("e-mail") and "chat rooms" or "CB simulators" by which persons remote from each other can send and receive more or less personalized messages to each other, depending upon the context. Entities offering these ancillary services, in addition to data and information access and retrieval, are generically referred to as on-line services. Perhaps the best known on-line service of the 1980's and 1990's is CompuServe.®

Until recently, whether those seeking data did so through on-line services, such as CompuServe, or through proprietary data and information storage and retrieval services such as

Lexis,® Dialog, or Westlaw,® access invariably was achieved through one of two means: (i) generic ASCII or ANSI communications or (ii) proprietary access. Each such approach offered advantages, but each also had drawbacks.

With generic access, users could access the on-line or information providers' services through virtually any generally available terminal program. Such terminal programs included Procomm Plus® and Q-Modem.® Using these programs, the user could issue commands from his/her terminal to a program running on one or more host computers located at the on-line services' central base of operations. In so-called expert mode, users were presented, at their terminals, with little more than a blank screen or single line prompt. The user was required to type in commands like "GO MAIL" or "GO STOCK."

In more user friendly modes, the information provider's host program would send a series of messages for display on the user's terminal. These messages would include a list of options with one of several prescribed responses. For example, the user's terminal screen might look something like the following:

CHOOSE THE APPROPRIATE OPTION FROM THE LIST BELOW

1. Send and receive e-mail messages
2. News, weather, and sports
3. Travel and entertainment
4. Computers and technology
5. Classified
6. Reference library
7. Free software (uploads and downloads)
8. Stock market
9. Sign off

Your command please:

The user could select one of the options by typing in the number corresponding to the option and by pressing the ENTER or RETURN key. Optionally, one could enter the expert command, such as "GO MAIL" to go to the e-mail portion of the service.

The principal advantage of this access method was that access was available to anyone with a terminal program, no matter how unsophisticated. There was no need for specialized terminals or specialized software. The information providers did not have to educate users on the special commands and methods of operation for the access software and also educate them on the commands required to use the programs residing on their resident host computers. Information providers also did not have to bear the cost of developing proprietary access software.

The principal disadvantage of these systems was their rigid, text-based displays and their complexity. One either had to know a range of complex expert commands or wait for a screen full of explanation to scroll to the screen. This information would come in serial form, one line after another. As new information was provided, the information previously available was scrolled up and/or off the screen. Not only was this process slow, but it was difficult to make sense of the instructions and explanations being provided when only a portion was visible at one time. The text-based displays were also uninteresting and unappealing, dissuading all but the most determined from joining into the information revolution.

In an effort to make their services easier to use and to make graphics available, information providers started offering proprietary access software programs that could be executed on the users' personal computers. However, there is no uniformity among such computer programs. As a result, users have to learn entirely different programs in order to access, for example, Dialog,® Lexis, Westlaw,® CompuServe,® and, more recently, America Online.®

The costs associated with offering these proprietary access programs also present a problem. The information providers have to engage the services of programmers to write the access programs. Technical support must be made available to assist new users in loading and using the software and in dealing with incompatibility problems. These costs are even further aggravated by the need to develop products for the various operating systems running on personal computers, e.g., Microsoft DOS,® Microsoft Windows,® Macintosh Finder,® and Unix.®

Finally, there is the problem and cost of distribution. Anyone who has purchased a personal computer in the last half decade has either seen icons for CompuServe® and America Online pre-loaded on his/her computer, or has received, in the mail, an endless supply of floppy disks with the information providers' "free" access software. These development, maintenance, support and distribution costs are then passed on to customers/users in their subscription charges.

At about the same time these competing approaches to information access were vying for acceptance in the general commercial marketplace, an entirely different system was gaining acceptance in government and academic circles. Professors and students were e-mailing each other and accessing remote databases over what was and is called the Internet.®

The Internet® is a loosely organized system of interconnected computer systems that "talk" to each other through a defined protocol. This system was initially developed as a matter of national defense. Multiple interconnected computers with built-in redundancy, it was believed, would allow military response in the event nuclear attack disabled part of the system. Being confined to largely academic and military applications, this Unix-based system was itself difficult to use, being filled with arcane commands and concepts, like UUCP, TCIP, FTP, and an almost entirely text-based interface.

Half a decade ago two events coalesced to revolutionize data and information access. First, access to the Internet was opened up to the general public. Kids hardly out of college opened enterprises offering access or "on-ramps" to the Internet,® sometimes called the "Information Superhighway." These Internet® service providers ("ISP"s) gave virtually anyone and everyone access to the Internet.

For dial-up account users, the attractiveness of the Internet® was still somewhat limited. Users had to master the arcane Unix commands referenced above. Then came the second development that ignited the information access revolution. The World Wide Web ("WWW") and hyper-text markup language ("HTML") allowed users simply to "click" on pictures, fill out dialog boxes, navigate through check boxes, and otherwise use the features of the graphical user interfaces ("GUI"s) by then popular in Windows® and Macintosh® systems. Consequently, access to the vast, almost limitless resources of the Internet® and availability of GUI operation literally ignited an explosion of interest and demand for access to information over the Internet.

Nowhere was the impact of the Internet® more apparent than upon the fortunes of a group of graduate students who commercialized a product they developed as a research project. Netscape Navigator® became an instant success and made these students instant millionaires. Netscape Navigator,® a browser, interfaces with the Internet® to make all the advantages of the GUI available over the Internet.® Soon, along came Internet Explorer,® another browser, from Microsoft,® and the rest is daily news. Easy and pervasive access was available through these browsers, and information providers no longer had to develop and market their proprietary access software packages to permit access to their systems.

The WWW has become ubiquitous. Companies communicate with each other across the country and around the globe using the Internet.® Businesses offering goods and services for sale advertise on the WWW. More recently, electronic commerce, once confined to proprietary network providers of electronic data interchange ("EDI"), now takes place over the WWW. Stocks, interests in mutual funds, airline tickets, music, software, and even real estate can be purchased and sold over the WWW. Almost limitless data and information resources are available over the WWW. And, to an increasing degree, access to proprietary databases such as Dialog® is now available through the WWW, and all major information and service providers are developing Web-based access to their systems.

With access to the WWW through browsers so prevalent, it is of enormous commercial benefit for those in need of remote communications or data access to use the WWW. First, one avoids the high cost of maintaining a private network. To the extent information consumers are willing to pay themselves for access to the Internet, providers of goods and services can simply make those goods and services available over the Internet,® without having to deal with the complexities and cost of their own telecommunications networks. Second, to the extent users are already familiar with their own browser

software, there is no need to provide proprietary software. Consequently, the WWW has brought together the benefit of universal access, once available only through the plain ASCII or ANSI interface of the past, and the user friendly graphical characteristics, once available only through proprietary data access software.

Some problems still remain, however. First, to date, all interconnection between a local terminal and the WWW relies on the browser or so-called software plug-ins imbedded in the browser. These plug-ins sit on top of the browser and permit execution of files that are of a type other than that recognized by WWW protocol. Consequently, if one wants to use the WWW to interact with software or hardware residing on the terminal computer, one must tap into the functionality of the browser in use. This means such plug-ins must be developed, distributed, installed, and maintained for each of the different browser computer programs on the market, which, for the present, means principally Netscape Navigator® and Microsoft's Internet Explorer.® Redundant effort, therefore, is required.

This redundancy becomes even more significant when one considers that a software developer seeking to control local terminal functionality through means of browser plug-ins has no control over the extent to which programming changes in new releases of the browser may render his/her newly developed plug-in programs incompatible. Under such conditions, the costs and logistics of developing, distributing, installing and maintaining such plug-ins is hardly feasible. In addition, creativity suffers when programmers try to control local terminal functionality by means of browser plug-ins because the programmers are forced to depend upon the browser developers in terms of the code that is developed.

Second, in addition to the browser dependent problems referenced above, those seeking to control local terminal functionality such as scanners, printers, video cameras, fingerprint recognition devices, microphones, magnetic stripe or so-called "smart card" technology, or coin and/or cash receptors must do so from the local terminals and not the WWW site. This means, of course, that all functionality must be enabled prior to WWW access, not during WWW access and that it certainly is not intelligently enabled, i.e. based on interaction with the WWW site. As a result, there is no flexibility in dealing with a particular user.

For example, suppose a user accesses a web site from a home PC which is equipped with magnetic stripe and/or "smart card" technology. The user first desires to print out a series of documents retrieved from his search of a proprietary data base and pay the appropriate charges by swiping his magnetic stripe credit card. Next, this user wishes to withdraw cash from his checking account by updating his stored value "smart card" or debit card. Apart from some expensive-to-maintain local intelligence such as a plug-in or a proprietary browser, neither the Web site nor the browser "knows" that the user's terminal has these capabilities. Nor are they able to "turn on" devices such as card

readers/writers or fetch programs that walk the user through the appropriate series of screens in response to the available functionality and user input.

Examination of the idea of providing proprietary browser software to each local terminal to accommodate that terminal's unique on-board functionality reveals the third basic problem facing those seeking to control and/or enhance interaction between the WWW and local terminal functionality. As a result of controlling all functionality locally, all upgrades, enhancements, repairs or troubleshooting must occur at each site either by down-loading data to all access terminals or by physically distributing and installing the upgrades or performing repairs at each site instead of building in the commands or loading the data once at the server or Web site level. The time and other costs involved in such site-by-site maintenance activity are significant when dealing with a system of public communications terminal such as a functionally-rich kiosks, Automatic Teller Machines ("ATM"); such costs make the solution unfeasible when applied to the functionally-rich home computers of tomorrow. Further, even if there were a way to manage the costs, this need to broadcast data and code means that personnel in charge of the system must poll and verify each terminal, often interrupting local service -- not to mention the risk of human error through manual intervention and verification.

Finally, there is the problem of functionality optimization. Recognizing the benefits of the browser view on the world, systems have developed to add to the functionality of the Web sites. HTML has become more and more robust. In addition, systems like VB-Script, JAVA Script, and JAVA have made it possible to write software rivaling the functionality of computer programs written, for example, in C for the local terminal. The drawback is that it is difficult, if not impossible, to match the speed and efficiency of a local program optimized for the terminal's central processing unit ("CPU") chip in Web-based code running remotely. Thus one sacrifices speed, power, and efficiency in program execution by utilizing it through a window on the WWW.

Additionally, there is presently no way (outside a closed, proprietary system or proprietary browser) to "mix and match" Web-based dynamic application-level code with static system-level code to control and operate local functionality, thereby achieving maximum efficiency. If Web-based code is used, though the code could be centrally maintained and administered, it would have to be downloaded each time. As anyone who has accessed the Internet knows, one of the frustrations is "waiting" for Web sites to download. Even if, through compression techniques or other methods, download time could be significantly reduced, there is still the problem of having to download different versions of the Web-based code to activate and control different brands or models of functionally similar devices -- an awkward and inefficient proposition. If, however, Web code is not downloaded, local functionality would have to be controlled outside the browser, by proprietary browsers, proprietary browser plug-ins, or other local code that would have to be installed and updated by the user -- something that often does not happen or that happens incorrectly. If only the static system-level code

could continue to reside locally while the dynamic application-level code is stored and administered centrally (at the Web site) so that only the dynamic application-level code has to be downloaded, highly efficient Web-based control of local functionality would be possible.

Accordingly, it is a primary object of the present invention to provide a system that utilizes an entirely browser-independent conduit to the WWW to activate and control functionality residing at the local data access terminal.

It is another object of the invention, to permit local functionality, in terms of hardware and/or software, to be controlled from, and in response to, Web site activity, including both data and code residing at the Web site and user input.

It is yet another object of the invention to permit code and data to be made available for controlling local functionality for terminals accessing a Web site without separately polling and/or downloading such code and data to local access terminals.

It is another object of the invention to provide a system where local functionality can be controlled using the same static level system code either through the browser or through freestanding local applications;

It is still another object of the invention to combine the benefits of the ubiquitous WWW and the familiar interface to the WWW with the speed, power, and efficiency of hardware and software developed and, quite possibly, optimized for the CPU chip in the local access terminal.

It is another object of the invention to optimize efficiency by permitting local functionality to be operated and controlled by static system-level code that continues to reside locally so that only the dynamic application-level code need be downloaded by the browser upon interconnection.

SUMMARY OF THE INVENTION

The present invention relates to a unique system and method for enabling a Web site to communicate with and optionally activate and/or control hardware and software located on the access terminal without having to depend upon or work with a particular browser. This method and system consists of a unique applications program interface ("API") which acts as a go between, translator, and/or message carrier between a standard Internet browser and local system-level code.

Utilizing one of a number of protocols for Web-based functionality such as JAVA or Active-X, this API is triggered, not by a user-based identification code, but by the very establishment of the interconnection between the local terminal and the Web-based code, regardless of where such code resides, e.g., on a remote Web site over the Internet, within an organization's intranet, or on the terminal itself. For purposes of simplicity, throughout the remainder of this application, discussion will focus primarily upon the use of the invention where the dynamic application-level code resides on a remote Web site accessed over the Internet® (rather than on a more or less local intranet site or on the terminal itself) because that is where it is believed the principal utility of the invention lies. (The term "Web site," however, is to be interpreted and defined herein as including a remote web site over the Internet,® a web site within an organization's intranet, an html web site residing on the accessing terminal itself, and all equivalents thereof.) The Web site and the local terminal, through the API, without the user's knowledge or intervention, determine what functionality is available and then, depending upon programming choices, activates and or controls the available functionality from the Web site, not the local terminal.

Through one of several means, the local terminal and the Web site introduce each other. Code residing locally may, through the API, signal the Web site that the local terminal has certain functionality. This is accomplished through the browser's interconnection, but not through the browser's code or the code of a browser plug-in. Based upon that introduction, the Web site makes a decision as to what if any screens are to be made available. The Web site may request user input such as the entry of an identifying code. It may take entirely user-independent action such as turning on a concealed video/audio monitoring device. It may cause a combination of activities, e.g., prompt the user to place his/her magnetic card or so-called "smart card" in the card reader's slot, turn on the card reader, activate the read head, upload the information, and prompt the user to proceed.

In an alternative embodiment, the introduction can come from the Web site. For example, the Web site, upon interconnection and using the API, can "check" to see if the local terminal bears one or more "signatures." "Signatures," herein, are data, code, objects or any other means of sending information and all equivalents thereof. Depending upon what signatures are found, different functionality can be enabled, turned-on, or accessed. For example, if none of the signatures is found,

the Web site can "block" further access. In the alternative, it may start a dialog with the user or initiate other actions, as described above.

The API, for these purposes, would reside at the local terminal. In an alternative embodiment, upon interconnection, the Web site could immediately send the API to the local terminal at which point the Web site could use the API to search for "signatures" or alternatively the code at the local terminal could signal the Web site through the API.

Once the initial introductions are made, the local terminal and the remote Web site can carry on what amounts to a three-way conversation, to use a metaphor. At each level of interaction intelligent decisions can be made at the Web site as to what actions are appropriate or permitted, based upon the user's identity, the terminal being used, and the interaction between the user and the Web site.

Based on the user's identity and the terminal, the Web site knows what screens to make available. For example, if the Web site, in the introduction phase, determines that the terminal is an intelligent, feature-rich public access kiosk, the Web site might present the user with a range of choices, including, scanning a document, choosing a means of payment, making a voice telephone call, or accessing a local program such as Microsoft Word.® If, in the introduction phase, the Web site, through the API, determines that the access terminal is a personal computer equipped with a magnetic stripe scanner or a smart card reader, it may prompt the user to swipe or insert his/her credit card or smart card. If, however, the Web site, through the API detects that the access terminal is an ordinary personal computer, it may prompt the user to input (manually type-in) his/her credit card number and expiration date or send the user down an entirely different access route.

As noted earlier, the API uses standard Web technology protocol. Instead of using it for its intended purpose, creating functionality or even computer programs executing at the Web site, it quietly nestles between the local access terminal and the Web site taking instructions from both the Web site, through a combination of programmed choices and user input, and from the local terminal, depending upon the hardware and software connected with or loaded on the local terminal.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG.1 is a block diagram depicting the system and associated apparatus according to one embodiment of the invention wherein the API acts as the go-between for the Web site and local functionality by "talking" to local system manager software.

FIG. 2 is a block diagram depicting the system and associated apparatus according to one embodiment of the invention wherein the API is acting as a go-between for a Web site and an ordinary personal computer.

FIG. 3 is a block diagram depicting the system and associated apparatus according to one embodiment of the invention wherein the API is acting as a go-between a Web site and a functionally-rich personal computer.

Fig. 4 is a flow diagram illustrating the method, including the introduction process, according to one embodiment of the invention wherein the local terminal commences the introduction process by signaling the presence of special functionality.

Fig. 5 is a flow diagram showing the method according to one embodiment of the invention wherein the Web site initiates the introduction by "looking" for special signatures at the local terminal.

Fig. 6 is a flow diagram indicating the current options for accessing bank computer mainframes either by personal computer or by isolated ATM devices.

Fig. 7 is a flow diagram illustrating method whereby the present invention permits a mainframe computer functionality or a Web site functionality to be accessed remotely by an individual personal computer or by a dedicated distant site.

DESCRIPTION OF THE PRESENTLY PREFERRED EMBODIMENT

Reference is now made to the FIGS. wherein like parts or steps are referred to by like numerals throughout. Fig. 1 shows the basic components and relationship of the system and related apparatus of the current invention in a preferred embodiment *. The embodiment is preferred only in the sense that it more extensively utilizes the functionality of the invention.

In Fig. 1, the API 10 is shown as the go between for local system manager software 11 and a Web site 12. The system manager software 11 is a local program that acts as an extension of the local terminal's operating system to permit interaction with non-standard peripherals such as smart card technology 13, magnetic stripe readers 14, scanners 15, bill and coin receptacles 16, and video devices 17. The API 10 receives instructions from the Web site 12 and sends instructions to the Web site 12 through the browser's web page 18. These instructions may consist of data, code, or data and code together, with the latter most being referred to as "objects."

FIG. 2 illustrates yet another embodiment of the invention in which the API 20 serves as the go-between for the Web site 21 and an ordinary personal computer 22 having only standard peripheral functionality such as a printer 23.

FIG. 3 illustrates another embodiment of the invention wherein the API 30 is shown as the go-between for the Web site 31 and a personal computer 32 having several nonstandard functional capabilities such as "smart card" technology 33, magnetic card reader/writer 34, and finger print scanner 35. In this embodiment, there is no local system manager software (FIG. 1, 11) *per se*, but merely the static system-level programs to run the on-board hardware (FIG. 3, 33-35).

Referring now to FIGS. 4 and 5, once the browser 40 has contacted the Web site 41 through an introduction process (shown generally at 42), the API 43 communicates to the Web page 44 details about the accessing terminal 45. This introduction process 42 occurs in one of at least two ways: 1) the API 43, already on the accessing terminal 45, introduces and makes itself known to the Web site 41 by sending one or more objects to the Web site 41 through the browser 40; or 2) the Web site (FIG. 5; 50), through the browser 51 and the web page 52 (again running on the local access terminal 53), can poll the local access terminal 53, to determine the presence of "signatures" indicating certain functionality, which are communicated to the browser's Web page 52, and hence the Web site 54 through the API 55. In either case, communication with the local terminal (45, 53) occurs, not through the browser's (40, 51) code or functionality or the code or functionality of a browser plug-in, but through the API (43, 55).

The API is able to participate in this "introduction" process because the API utilizes Web protocol intended for Web-based programming to obtain a new, different, and unexpected result. Instead of permitting robust access to functionality at the Web server through the browser -- the purpose of

systems like JAVA and Active-X, the present invention enables functionality at the local terminal, not just the Web server. Thus, the functionality at the local terminal is controlled through the browser at the Web site level, not through separate code on the local terminal and not through code residing in the particular browser or any browser plug-in. The communication between the Web site and the local terminal's functionality, therefore, is truly browser independent.

Focusing again on FIG. 1, after serving an initial introductory function (see FIGS. 4 and 5), the API 10 serves as a go-between for the Web page (FIG. 1;18) and, in a preferred embodiment (FIG. 1), the local system manager software 11. Communication by the API 10, is made, therefore, not to the hardware or software on the local terminal 13-17 directly, but through local system manager software 11. The local system manager software 11 was created as an extension of standard personal computer operating system functionality to take advantage of hardware and software not ordinarily found on personal computers, but essential, for example, in the public access terminal industry. Thus, the API communicates with the local system manager software which, in turn, communicates with bar code readers, magnetic stripe readers, scanners, video cameras, fingerprint recognition devices, smart card readers, and other devices.

The Web site, depending upon the information learned about the accessing terminal during the introduction phase, may either send objects to the API for communication to the local system manager software immediately or may communicate, through the browser, with the user. The Web page may then further communicate with the API , and hence, the local system manager software, by sending objects consisting both of code from the Web page and data, either from the Web site itself or from user input. In the latter case, user input, such as commands, though entered at the local terminal, are not directly processed by the local terminal. Instead, such input is communicated, through the browser, to the Web site. Such input, coupled with code and/or other data, is then communicated to the hardware or software on the user's terminal through the API.

The Web site, therefore, can make intelligent decisions. If it is "introduced" to a functionally intelligent terminal, such as that of FIG. 1 or FIG. 3, it can capitalize on that functionality by providing various options to the user. If, on the other hand, the "introductions" reveal that the accessing terminal is an ordinary personal computer with only standardized functionality (FIG. 2), the user will be directed, for instance, to manually enter their credit card number and expiration date.

In addition to identifying the existence or nonexistence of functionality at the local terminal, the present invention can also relay information that the a particular functional device is inoperative and initiate programs to circumvent the problem. For instance, the Web site (FIG. 1; 12), as an alternative, could prompt the user to type in his/her credit card number manually if the local system manager software 11 detected a problem with the magnetic stripe reader 14. The local system manager software 11, in this case, would attempt to turn on the reader 14. In case of malfunction, that

fact would be communicated to the API 10. The API 10, in turn, would deliver that message to the Web site 12. The Web site 12 could then, based upon that message, inform the user that the reader 14 is presently inoperable and prompt him/her to enter manually his/her credit card number.

The invention can be best described by a concrete example, though the scope of the invention cannot be limited to the circumstances of the specific examples. Mr. Citizen decides it is time to take care of a few matters he has been putting off and proceeds to the nearby mall where a Government Services Transaction Kiosk is located. Among the tasks Mr. Citizen would like to accomplish are renewing his driver's license, paying a traffic ticket, and checking the status of his personal property taxes.

He swipes the magnetic stripe of his expired license, which activates local communication software and hardware. Mr. Citizen waits but a moment as a communication link is established first with an Internet Service Provider ("ISP") and then with the Government's Web site. The data read from the card is then passed by the local system manager software to the API. The API tells the Web site who is accessing the terminal based on the card's data. At that point, the Web page (not the local machine) can access its own centralized intelligence to determine what, if any, additional verifications such as passwords, access codes, fingerprint identification, voice prints, or the like are required. Depending upon the programming at the Web page, the next step is determined. The "view" Mr. Citizen perceives is essentially the same he would receive at home, using his own personal computer, with identical commands and an identical user interface. The differences, as will be discussed further, are related solely to the contents of the Web site, which differences are the direct result of the availability of functions at the specific terminal from which access is achieved.

As the browser page appears to Mr. Citizen on the kiosk's touchscreen, the API, residing on the kiosk, is signaling the Web site about the functional characteristics of the kiosk by sending data, code, or objects indicating that the accessing terminal is a registered government kiosk with a plethora of functionality. An application program at the Web site is triggered which then sends a menu list of service selections to Mr. Citizen through the browser. For example:

Please choose your service from the selection below by touching
your selection on the screen:

1) Renew your driver's license.
2) Renew your vehicle license.
3) Get an estimate of personal property taxes.
4) Pay your personal property taxes.
5) Enter a plea and/or Pay your traffic fines.
6) Exit

Mr. Citizen selects "1," and is presented with a set of instructional steps which first lead him through various security and identification processes necessary for completing this transaction such as entering his social security number and placing his finger in a fingerprint identification reader. The Web site, being aware that the kiosk has these functional features, fetches the appropriate instructional programs and "turns on and off" the functionality on the kiosk at the appropriate times. Had Mr. Citizen accessed the Web site through his own personal computer, he would have been presented with a different screen that would have had some, but not all, of the options presented. Moreover, the prompt would likely have said "Click" on the desired selection instead of "touch" it.

To insure that Mr. Citizen is the person identified in the expired license records, the Web site will fetch and activate application programs for taking Mr. Citizen's picture via the video camera at the kiosk and then printing and molding the driver's license for delivery on site to Mr. Citizen. At each step, the API communicates data to the Web site via the browser. The Web site then fetches appropriate commands, sends them through the browser to the API, which then enables and/or disables the appropriate hardware.

By the same process, Mr. Citizen may select services enabling him to enter a plea on his speeding ticket by signing a digital script pad, which signature is then transmitted over an Internet® provider network. He may then pay the ticket either by swiping his magnetic stripe credit or debit card, withdrawing cash from his "smart card," or depositing into bill and coin receptors. Mr. Citizen may also choose to access his personal property tax records, print a copy, decide that the recorded balance is wrong, and initiate a video conference call to discuss the disputed amount with the appropriate office, charging the call and printing to his magnetic stripe card.

All along the way in each of these examples the Web site is savvy to Mr. Citizen's choices. For example, when he signs the digital script pad, the API sends a signal to the Web site through the browser, data is sent and received, and applications software is fetched and triggered to guide him to the next appropriate step. When Mr. Citizen swipes his magnetic stripe card, the API sends a signal to the Web site through the browser's Web page and the Web site can then return a message telling the API to "turn on the reader" and fetch application programs which provide appropriate screens to guide Mr. Citizen through to the next step of the process.

As is revealed the system and method disclosed are flexible and responsive to the activities of the user and the functional capabilities of the local terminal. The Web site knows when it has a functionally-rich site such as the kiosk in FIG. 1 and the example above. It also will know when the access terminal is a functionally "dumb" terminal such as the one depicted in FIG. 2. In this case, no special hardware or software is detected. Accordingly, the API signals the Web site to direct the user down a route far different from that available to the user accessing the Web site from the terminal depicted in Fig. 1.

If, on the other hand, a user has a personal computer that is equipped with "smart card" technology (FIG. 3), he could conveniently use the method and system of the present invention to withdraw or deposit cash from or to his checking account 24 hours a day without ever leaving home. According to this embodiment of the invention, the user's home computer would access a bank's Web site. Upon connection, the local PC would either send signals through the API that it is equipped with this functionality or the bank's Web would search for the functionality on the local PC through the API. This API would either be resident at the local PC already, or could be sent there from the Web site immediately upon connection. Once the functionality introductions were complete, various security checks could be initiated, and eventually the Web site would direct the smart card reader/writer to perform the appropriate functions.

In today's world, if Mr. Citizen wants cash he likely will go to his bank, write a check for cash, or go to a networked public Automatic Teller Machine (ATM) (61, FIG. 6). These ATM networks are highly-controlled proprietary systems wherein a bank's central mainframe 60 communicates directly with each ATM 61. The programming at the central mainframe 60 is designed and configured for the functionality on-board the ATM 61 and the ATM 61, itself, is programmed to operate within this closed system. Recently, some banks have begun to offer access to banking services through personal computers (PCS) 62. To do so has required development of specialized software, distribution of that software to each individual user, and resulted in all the typical customer service difficulties that ensue when numerous individuals are installing and operating software independently. In other words, the bank is having to develop, operate, and service two entirely separate systems.

The present invention allows the same functionality to be made available both to feature and function rich public access terminals (FIG. 7; 70), such as ATMs, and to ordinary personal computers 71 through one system via the API 73. There is no longer any need to develop, distribute, and install specialized software or to operate two entirely separate systems for access to such services and functionality. The home PC of tomorrow will no doubt include functionality such as smart and magnetic card technology, script pads, and the like. According to the present invention, a bank through its mainframe 72 could offer access to Mr. Citizen at an ATM 70 where he could withdraw cold, hard cash or update his stored value or smart card as he desired; or, in the alternative, and through the same system, the bank could recognize, enable and respond to the functionality available on Mr. Citizen's home PC 71, updating his value added or debit card.

Similarly, a user with a local PC, equipped as described above or with a magnetic stripe reader/writer, who contacts the WWW through the user's regular browser, may purchase copies of copyrighted documents from proprietary data bases and/or order merchandise from other Web sites and pay on the spot through the activation of their smart card or magnetic stripe card reading/writing devices. Not only is this a convenience, but because the user has actually submitted his/her card rather

than merely inputting his/her card information into a data box, the transaction is more secure and therefore entitled to a better transactional rate.

In addition to the advantages the invention provides to the user, there are also distinct advantages to businesses offering services over the Internet. In each of these examples, the process is controlled largely from the central location, the Web site, permitting automated maintenance and centralized management of the application programs involved and automated maintenance and management of a whole system of public access computer kiosks. The Web site, in these examples, can make intelligent decisions.

Following is a print-out of representative source code for an API in accordance with one embodiment of the invention:

```
// TNMagStripeCtl.cpp : Implementation of the CTNMagStripeCtrl ActiveX Control class.
/*
// The control has several properties and methods.  Anything in the Web Space
// can call the methods or change the properties.  The Control also has several
// events defined that it will fire out in to the web space.
 The Properties:
-------------------------------------------------------------------------
Web Space Name   ||  Internal Control Name  ||  Brief Description
-------------------------------------------------------------------------
PINRequired      m_pINRequired          'TRUE' if PIN is req. by card
CCFormatCode     m_cCFormatCode         Format Code from the Card
CCPAN            m_cCPAN                PAN from the Card
CCName           m_cCName               Person's name from the Card
CCExpDate        m_cCExpDate            Expire Date on Card
CCServiceCode    m_cCServiceCode        Service Code on Card
CCAddlData       m_cCAddlData           Addl Data field ON Card
CCIssuerCode     m_cCIssuerCode         Issue date of Card
CCIssuerName     m_cCIssuerName         Issuer Name on Card
CardSuccess      m_cardSuccess          'TRUE' if card has been validated
    CCTrack1          m_cCTrack1          Raw Track 1 data from card
    CCTrack2          m_cCTrack2          Raw Track 2 data from card
    CCTrack3          m_cCTrack3          Raw Track 3 data from card
    CCTrackNumber     m_cCTrackNumber     Which track holds the PAN
```

-- the following three are accessed with Get/Set methods
to write-protect them

MajorVersion	These are version numbers for
MinorVersion	feature checking. As the version
BuildNumber	numbers increase, more features will
	appear. . .

The Methods:

Web Space Name	Brief Description
InitControl	Initializes Control, must be called before control will work
DeInitControl	DeInits the control, must be called for overall kiosk to work
EnablePayment	Turns on Payment functionality (accept credit card, coins, etc.)
DisablePayment	Turns off Payment functionality
UsePaymentCardTable	Use the PaymentCard table to validate Cards
UseIDCardTable	Use the IDCard table to validate Cards
EnableCardReader	Turn on Card Reader functions only (do not accept coins)

The Events:

Web Space Name	Brief Description
CardData	Signifies Card has been at least partially validated
CardInserted	Signifies Card has been inserted into card reader
CardError	Signifies a card error has occurred (bad card, failed validation)
CardDisable	Signifies Card reader has been disabled
CardEnable	Signifies Card reader has been enabled
CardReset	Signifies Card reader has been reset
CardTest	Signifies Card reader is being tested
CardInvalid	Signifies Card is Invalid

*/

```
#include "stdafx.h"
#include "TNMagStripe.h"
#include "TNMagStripeCtl.h"
#include "TNMagStripePpg.h"
```

```
// TouchNet header files. . .
#include <tntypes.h>
#include <tntools.h>
#include <tnapi.h>
#include <tnmsg.h>
#include <msgunit.h>
#include <tnrmi.h>
#include <tncommon.h>
#include <tnprn_hk.h>
#include <tnbtnsnd.h>
#include <tnhooks.h>
#include <tnpmmsgs.h>
#include <tnlaunch.h>
#include <time.h>
#include <tndir.h>
#include "version.c"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
IMPLEMENT_DYNCREATE(CTNMagStripeCtrl, COleControl)
////////////////////////////////////////////////////////////////////////////
// Message map
BEGIN_MESSAGE_MAP(CTNMagStripeCtrl, COleControl)
  ON_MESSAGE(TNMSG_MESSAGE_FROM_ENGINE,OnMessageFromEngine)
      //{{AFX_MSG_MAP(CTNMagStripeCtrl)
      // NOTE - ClassWizard will add and remove message map entries
      //    DO NOT EDIT what you see in these blocks of generated code !
      //}}AFX_MSG_MAP
      ON_OLEVERB(AFX_IDS_VERB_PROPERTIES, OnProperties)
END_MESSAGE_MAP()
////////////////////////////////////////////////////////////////////////////
```

```
// Dispatch map
BEGIN_DISPATCH_MAP(CTNMagStripeCtrl, COleControl)
        //{{AFX_DISPATCH_MAP(CTNMagStripeCtrl)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "PINRequired", m_pINRequired,
                OnPINRequiredChanged, VT_BOOL)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CCFormatCode", m_cCFormatCode,
                OnCCFormatCodeChanged, VT_BSTR)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CCPAN", m_cCPAN,
OnCCPANChanged, VT_BSTR)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CCName", m_cCName,
OnCCNameChanged, VT_BSTR)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CCExpDate", m_cCExpDate,
OnCCExpDateChanged,
                VT_BSTR)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CCServiceCode", m_cCServiceCode,
                OnCCServiceCodeChanged, VT_BSTR)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CCAddlData", m_cCAddlData,
                OnCCAddlDataChanged, VT_BSTR)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CCIssuerCode", m_cCIssuerCode,
                OnCCIssuerCodeChanged, VT_BSTR)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CCIssuerName", m_cCIssuerName,
                OnCCIssuerNameChanged, VT_BSTR)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CardSuccess", m_cardSuccess,
                OnCardSuccessChanged, VT_BOOL)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CCTrack1", m_cCTrack1,
                OnCCTrack1Changed, VT_BSTR)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CCTrack2", m_cCTrack2,
                OnCCTrack2Changed, VT_BSTR)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CCTrack3", m_cCTrack3,
                OnCCTrack3Changed, VT_BSTR)
        DISP_PROPERTY_NOTIFY(CTNMagStripeCtrl, "CCTrackNumber",
                m_cCTrackNumber, OnCCTrackNumberChanged, VT_I2)
        DISP_PROPERTY_EX(CTNMagStripeCtrl, "MajorVersion", GetMajorVersion,
                SetMajorVersion, VT_I2)
        DISP_PROPERTY_EX(CTNMagStripeCtrl, "MinorVersion", GetMinorVersion,
```

```
        SetMinorVersion, VT_I2)
    DISP_PROPERTY_EX(CTNMagStripeCtrl, "BuildNumber", GetBuildNumber,
        SetBuildNumber, VT_I2)
    DISP_FUNCTION(CTNMagStripeCtrl, "InitControl", InitControl, VT_I2, VTS_NONE)
    DISP_FUNCTION(CTNMagStripeCtrl, "DeInitControl", DeInitControl, VT_I2,
        VTS_NONE)
    DISP_FUNCTION(CTNMagStripeCtrl, "EnablePayment", EnablePayment, VT_I2,
        VTS_NONE)
    DISP_FUNCTION(CTNMagStripeCtrl, "DisablePayment", DisablePayment, VT_I2,
        VTS_NONE)
    DISP_FUNCTION(CTNMagStripeCtrl, "UsePaymentCardTable", UsePaymentCardTable,
        VT_I2, VTS_NONE)
    DISP_FUNCTION(CTNMagStripeCtrl, "UseIDCardTable", UseIDCardTable, VT_I2,
        VTS_NONE)
    DISP_FUNCTION(CTNMagStripeCtrl, "EnableCardReader", EnableCardReader,
        VT_I2, VTS_NONE)
    //}}AFX_DISPATCH_MAP
    DISP_FUNCTION_ID(CTNMagStripeCtrl, "AboutBox", DISPID_ABOUTBOX,
AboutBox,
        VT_EMPTY, VTS_NONE)
END_DISPATCH_MAP()
////////////////////////////////////////////////////////////////////////////
// Event map
// TouchNet defines 'new' events to 'fire' out to the web browser or whatever
// container includes the control.
BEGIN_EVENT_MAP(CTNMagStripeCtrl, COleControl)
    //{{AFX_EVENT_MAP(CTNMagStripeCtrl)
    EVENT_CUSTOM("CardData", FireCardData, VTS_NONE)
    EVENT_CUSTOM("CardInserted", FireCardInserted, VTS_NONE)
    EVENT_CUSTOM("CardError", FireCardError, VTS_NONE)
    EVENT_CUSTOM("CardDisable", FireCardDisable, VTS_NONE)
    EVENT_CUSTOM("CardEnable", FireCardEnable, VTS_NONE)
    EVENT_CUSTOM("CardReset", FireCardReset, VTS_NONE)
    EVENT_CUSTOM("CardTest", FireCardTest, VTS_NONE)
    EVENT_CUSTOM("CardInvalid", FireCardInvalid, VTS_NONE)
```

```
    //}}AFX_EVENT_MAP
END_EVENT_MAP()
////////////////////////////////////////////////////////////////////////////
// Property pages
// TODO: Add more property pages as needed.  Remember to increase the count!
BEGIN_PROPPAGEIDS(CTNMagStripeCtrl, 1)
        PROPPAGEID(CTNMagStripePropPage::guid)
END_PROPPAGEIDS(CTNMagStripeCtrl)
// TouchNet Added IObjectSafety Interface for Scripting capability
// Added for IObjectSafe Interface
////////////////////////////////////////////////////////////////////////////
// Interface map for IObjectSafety
BEGIN_INTERFACE_MAP( CTNMagStripeCtrl, COleControl )
  INTERFACE_PART(CTNMagStripeCtrl, IID_IObjectSafety, ObjSafe)
END_INTERFACE_MAP()
////////////////////////////////////////////////////////////////////////////
// IObjectSafety member functions
// Delegate AddRef, Release, QueryInterface
ULONG FAR EXPORT CTNMagStripeCtrl::XObjSafe::AddRef()
{
   METHOD_PROLOGUE(CTNMagStripeCtrl, ObjSafe)
   return pThis->ExternalAddRef();
}
ULONG FAR EXPORT CTNMagStripeCtrl::XObjSafe::Release()
{
   METHOD_PROLOGUE(CTNMagStripeCtrl, ObjSafe)
   return pThis->ExternalRelease();
}
HRESULT FAR EXPORT CTNMagStripeCtrl::XObjSafe::QueryInterface(
   REFIID iid, void FAR* FAR* ppvObj)
{
   METHOD_PROLOGUE(CTNMagStripeCtrl, ObjSafe)
   return (HRESULT)pThis->ExternalQueryInterface(&iid, ppvObj);
}
```

```
const DWORD dwSupportedBits =
    INTERFACESAFE_FOR_UNTRUSTED_CALLER |
    INTERFACESAFE_FOR_UNTRUSTED_DATA;
const DWORD dwNotSupportedBits = ~ dwSupportedBits;
////////////////////////////////////////////////////////////////////////
// CTNMagStripeCtrl::XObjSafe::GetInterfaceSafetyOptions
// Allows container to query what interfaces are safe for what. We're
// optimizing significantly by ignoring which interface the caller is
// asking for.
HRESULT STDMETHODCALLTYPE
  CTNMagStripeCtrl::XObjSafe::GetInterfaceSafetyOptions(
    /* [in] */ REFIID riid,
      /* [out] */ DWORD __RPC_FAR *pdwSupportedOptions,
      /* [out] */ DWORD __RPC_FAR *pdwEnabledOptions)
{
  METHOD_PROLOGUE(CTNMagStripeCtrl, ObjSafe)
  HRESULT retval = ResultFromScode(S_OK);
  // does interface exist?
  IUnknown FAR* punkInterface;
  retval = pThis->ExternalQueryInterface(&riid,
          (void * *)&punkInterface);
  if (retval != E_NOINTERFACE) { // interface exists
    punkInterface->Release(); // release it-just checking!
  }
  // we support both kinds of safety and have always both set,
  // regardless of interface
  *pdwSupportedOptions = *pdwEnabledOptions = dwSupportedBits;
  return retval; // E_NOINTERFACE if QI failed
}
////////////////////////////////////////////////////////////////////////
// CTNMagStripeCtrl::XObjSafe::SetInterfaceSafetyOptions
// Since we're always safe, this is a no-brainer, but we do check to make
// sure the interface requested exists and that the options we're asked to
// set exist and are set on (we don't support unsafe mode).
```

```
HRESULT STDMETHODCALLTYPE
  CTNMagStripeCtrl::XObjSafe::SetInterfaceSafetyOptions(
      /* [in] */ REFIID riid,
      /* [in] */ DWORD dwOptionSetMask,
      /* [in] */ DWORD dwEnabledOptions)
{
   METHOD_PROLOGUE(CTNMagStripeCtrl, ObjSafe)
  // does interface exist?
  IUnknown FAR* punkInterface;
  pThis->ExternalQueryInterface(&riid, (void * *)&punkInterface);
  if (punkInterface) { // interface exists
    punkInterface->Release(); // release it-just checking!
  }
  else { // interface doesn't exist
    return ResultFromScode(E_NOINTERFACE);
  }
  // can't set bits we don't support
  if (dwOptionSetMask & dwNotSupportedBits) {
    return ResultFromScode(E_FAIL);
  }
  // can't set bits we do support to zero
  dwEnabledOptions &= dwSupportedBits;
  // (we already know there are no extra bits in mask )
  if ((dwOptionSetMask & dwEnabledOptions) !=
      dwOptionSetMask) {
    return ResultFromScode(E_FAIL);
  }
// don't need to change anything since we're always safe
return ResultFromScode(S_OK);
}
// Added for IObjectSafe Interface End
////////////////////////////////////////////////////////////////////////////
// Initialize class factory and guid
IMPLEMENT_OLECREATE_EX(CTNMagStripeCtrl, "TNMAGSTRIPE.TNMagStripeCtrl.1",
        0x5b4ee8b8, 0xde71, 0x11d0, 0xa6, 0xd2, 0, 0x60, 0x97, 0xd2, 0x14, 0xc6)
```

```
////////////////////////////////////////////////////////////////////////////
// Type library ID and version
IMPLEMENT_OLETYPELIB(CTNMagStripeCtrl, _tlid, _wVerMajor, _wVerMinor)
////////////////////////////////////////////////////////////////////////////
// Interface IDs
const IID BASED_CODE IID_DTNMagStripe =
        { 0x5b4ee8b6, 0xde71, 0x11d0, { 0xa6, 0xd2, 0, 0x60, 0x97, 0xd2,0x14, 0xc6 } };
const IID BASED_CODE IID_DTNMagStripeEvents =
        { 0x5b4ee8b7, 0xde71, 0x11d0, { 0xa6, 0xd2, 0, 0x60, 0x97, 0xd2, 0x14, 0xc6 } };
////////////////////////////////////////////////////////////////////////////
// Control type information
static const DWORD BASED_CODE _dwTNMagStripeOleMisc =
        OLEMISC_ACTIVATEWHENVISIBLE |
        OLEMISC_SETCLIENTSITEFIRST |
        OLEMISC_INSIDEOUT |
        OLEMISC_CANTLINKINSIDE |
        OLEMISC_RECOMPOSEONRESIZE;
IMPLEMENT_OLECTLTYPE(CTNMagStripeCtrl, IDS_TNMAGSTRIPE,
_dwTNMagStripeOleMisc)
////////////////////////////////////////////////////////////////////////////
// CTNMagStripeCtrl::CTNMagStripeCtrlFactory::UpdateRegistry -
// Adds or removes system registry entries for CTNMagStripeCtrl
BOOL CTNMagStripeCtrl::CTNMagStripeCtrlFactory::UpdateRegistry(BOOL bRegister)
{
        // TODO: Verify that your control follows apartment-model threading rules.
        // Refer to MFC TechNote 64 for more information.
        // If your control does not conform to the apartment-model rules, then
        // you must modify the code below, changing the 6th parameter from
        // afxRegApartmentThreading to 0.
        if (bRegister)
                return AfxOleRegisterControlClass(
                        AfxGetInstanceHandle(),
                        m_clsid,
                        m_lpszProgID,
                        IDS_TNMAGSTRIPE,
```

```
                    IDB_TNMAGSTRIPE,
                    afxRegApartmentThreading,
                    _dwTNMagStripeOleMisc,
                    _tlid,
                    _wVerMajor,
                    _wVerMinor);
        else
                return AfxOleUnregisterClass(m_clsid, m_lpszProgID);
}
////////////////////////////////////////////////////////////////////////
// CTNMagStripeCtrl::CTNMagStripeCtrl - Constructor
CTNMagStripeCtrl::CTNMagStripeCtrl()
{
        InitializeIIDs(&IID_DTNMagStripe, &IID_DTNMagStripeEvents);
  SetInitialSize(0,0);
   // TouchNet added inits. . .
   pMsgUnit = NULL;
   pinNumber = "1234";
   fsFlags = 0;
   lBalance = 0L;
   // these numbers are not actually used (like the $2.50) they
   // just get the CreditCard API's going. . .
   lNeeded = 250L;
   minCreditNeeded = 100L;
   m_cardSuccess = FALSE;
   m_pINRequired = FALSE;
   m_cCPAN = "";
   m_cCFormatCode = "";
   m_cCName = "";
   m_cCExpDate = "";
   m_cCServiceCode = "";
   m_cCAddlData = "";
   m_cCIssuerCode = "";
   m_cCIssuerName = "";
   m_cCTrack1 = "";
```

```
  m_cCTrack2 = "";
  m_cCTrack3 = "";
  m_cCTrackNumber = 0;
}
////////////////////////////////////////////////////////////////////////////
// CTNMagStripeCtrl::~CTNMagStripeCtrl - Destructor
CTNMagStripeCtrl::~CTNMagStripeCtrl()
{
  // if the control is destructed, make sure to cleanup the Message Unit
  if(pMsgUnit)
  {
    tnMsgUnitDestroy(pMsgUnit);
  }
 }
////////////////////////////////////////////////////////////////////////////
// CTNMagStripeCtrl::OnDraw - Drawing function
void CTNMagStripeCtrl::OnDraw(
                    CDC* pdc, const CRect& rcBounds, const CRect& rcInvalid)
{
  // The control must technically be a 'visible' control in order to
  // be allocated a window handle.  But, we want it to be invisible
  // and allow the HTML (or other web-activated container) to take
  // the GUI, so we will draw nothing for the control.
  }
////////////////////////////////////////////////////////////////////////////
// CTNMagStripeCtrl::DoPropExchange - Persistence support
void CTNMagStripeCtrl::DoPropExchange(CPropExchange* pPX)
{
        ExchangeVersion(pPX, MAKELONG(_wVerMinor, _wVerMajor));
        COleControl::DoPropExchange(pPX);
}
////////////////////////////////////////////////////////////////////////////
// CTNMagStripeCtrl::GetControlFlags -
// Flags to customize MFC's implementation of ActiveX controls.
//
```

```
// For information on using these flags, please see MFC technical note
// #nnn, "Optimizing an ActiveX Control".
DWORD CTNMagStripeCtrl::GetControlFlags()
{
        DWORD dwFlags = COleControl::GetControlFlags();
        // The control will not be redrawn when making the transition
        // between the active and inactivate state.
        dwFlags |= noFlickerActivate;
        return dwFlags;
}
////////////////////////////////////////////////////////////////////////////
// CTNMagStripeCtrl::OnResetState - Reset control to default state
void CTNMagStripeCtrl::OnResetState()
{
        COleControl::OnResetState();  // Resets defaults found in DoPropExchange
}
////////////////////////////////////////////////////////////////////////////
// CTNMagStripeCtrl::AboutBox - Display an "About" box to the user
void CTNMagStripeCtrl::AboutBox()
{
        CDialog dlgAbout(IDD_ABOUTBOX_TNMAGSTRIPE);
        dlgAbout.DoModal();
}
////////////////////////////////////////////////////////////////////////////
// CTNMagStripeCtrl message handlers
////////////////////////////////////////////////////////////////////////////
LRESULT CTNMagStripeCtrl::OnMessageFromEngine(WPARAM wParam, LPARAM lParam)
{
  TNMSG_HEADER *pTNMsgHeader;
  int iRetVal;
  char szOutString[1000];
        // point the messge header to the address of the data
        pTNMsgHeader = (TNMSG_HEADER*)wParam;
  // When we get a message from the engine, we then do one of two
  // things:
```

```
//1) we just fire an event that goes directly into the HTML
// object modle so that it can be picked up by anything in the
// web space (JavaScript, VBScript, Dynamic HTML, etc. . . )
//2) we do some processing on the event (i.e. CardData) to see
// what event we want to fire into the web space.  The Engine
// sends a TNMSG_CARD_DATA message which the control, in turn,
// needs to validate and check against the Negative file, among
// other things in the future.
switch(pTNMsgHeader->msg)
{
  case TNMSG_CARD_DISABLE:
    m_cardSuccess = FALSE;
    FireCardDisable();
    break;
  case TNMSG_CARD_ENABLE:
    m_cardSuccess = FALSE;
    FireCardEnable();
    break;
  case TNMSG_CARD_RESET:
    m_cardSuccess = FALSE;
    FireCardReset();
    break;
  case TNMSG_CARD_ERROR:
    m_cardSuccess = FALSE;
    FireCardError();
    break;
  case TNMSG_CARD_TEST:
    m_cardSuccess = FALSE;
    FireCardTest();
    break;
  case TNMSG_CARD_INSERTED:
   m_cardSuccess = FALSE;
   FireCardInserted();
   break;
  case TNMSG_CARD_DATA:
```

```
{
  // several conclusions can come out of this case:
  // 1) A card can be accepted, no PIN required.
  // - the Property 'CardSuccess' will be set to TRUE
  // - the Property 'PINRequired' will be set to FALSE
  // - the 'CardData' Event will be fired
  ////////////////////////////////////////////////
  // 2) A card can be accepted, with a PIN required.
  // - the Property 'CardSuccess' will be set to TRUE
  // - the Property 'PINRequired' will be set to TRUE
  // - the 'CardData' Event will be fired
  ////////////////////////////////////////////////
  // 3) A card can be rejected for any number of reasons

  // - the Property 'CardSuccess' will be set to FALSE
  // - the Property 'PINRequired' will be ignored
  // - the 'CardInvalid' Event will be fired
     if(pTNMsgHeader->pData && pTNMsgHeader->DataLength == sizeof(RawData))
  {
    memmove(&RawData,pTNMsgHeader->pData,sizeof(RawData));
    tnMemSet(&ParsedData,0,sizeof(ParsedData));
// validate card will take the Raw Card Data and (if the card
    // is vallidated) will fill in the ParsedData fields
    // the ParsedData fields will also include items taken
    // form the CardTable which is individual for each machine.
    int rc = tnapiValidateCard(&RawData,&ParsedData);
    // if we need to check the Negative file (bad cards) and the card passes
    if(rc ||(ParsedData.fCheckNegFile && CheckNegFile(ParsedData.PAN)!=NO_ERROR))
    {
      // the card was in the Neg file and can't be used. .
      fprintf(stderr,"<<CARD INVALID>>\n");
      m_cardSuccess = FALSE;
      FireCardInvalid();
    }
    // and if the card is not expired. . .
```

```
else if(tnapiExpired(&ParsedData))
{
  // the card was expired already. . .
  fprintf(stderr,"<<CARD EXPIRED>>\n");
  m_cardSuccess = FALSE;
  FireCardInvalid();
}
else
{
  // if we need to get a pin,
  if(ParsedData.fPINRequired)
  {
    fprintf(stderr,"<<NEEDS PIN>>\n");
    // in this case we need to get the PIN number from
    // the user in order to properly use the card.
    // we will set the Data into the Properties so
    // that they will be in the Web Space in case anything
    // needs it for verification (last four digits of
    // card number, etc. . .
    m_cardSuccess = TRUE;
    m_pINRequired = TRUE;
    m_cCPAN = ParsedData.PAN;
    m_cCFormatCode = ParsedData.FormatCode;
    m_cCName =   ParsedData.Name;
    m_cCExpDate =   ParsedData.ExpDate;
    m_cCServiceCode =   ParsedData.ServiceCode;
    m_cCAddlData =   ParsedData.AddlData;
    m_cCIssuerCode =   ParsedData.IssuerCode;
    m_cCIssuerName =   ParsedData.IssuerName;
    m_cCTrack1 = RawData.Track1;
    m_cCTrack2 = RawData.Track2;
    m_cCTrack3 = RawData.Track3;
    if(strstr(RawData.Track1,ParsedData.PAN))
    {
      m_cCTrackNumber = 1;
```

```
    }
    else
    if(strstr(RawData.Track2,ParsedData.PAN))
    {
      m_cCTrackNumber = 2;
    }
    else
    if(strstr(RawData.Track3,ParsedData.PAN))
    {
      m_cCTrackNumber = 3;
    }
    FireCardData();
    break;
  }
  // if we already have a PIN, check it. . .
  if(!PinOK(pinNumber))
  {
    fprintf(stderr,"<<PIN BAD>>\n");
    m_cardSuccess = FALSE;
    FireCardInvalid();
    break;
  }
fprintf(stderr,"<<ABOUT TO AUTHORIZE CARD>>\n");
// payment must be disabled to authorize. . .
DisablePayment();
if(NO_ERROR==tnapiAuthorizeCard(&ParsedData,&RawData))
{
  // the card was accepted with no PIN required. . .
  m_cardSuccess = TRUE;
  m_pINRequired = FALSE;
  fprintf(stderr,"<<CARD ACCEPTED>>\n");
  m_cCPAN = ParsedData.PAN;
  m_cCFormatCode = ParsedData.FormatCode;
  m_cCName =   ParsedData.Name;
  m_cCExpDate =   ParsedData.ExpDate;
```

```
        m_cCServiceCode =  ParsedData.ServiceCode;
        m_cCAddlData =  ParsedData.AddlData;
        m_cCIssuerCode =  ParsedData.IssuerCode;
        m_cCIssuerName =  ParsedData.IssuerName;
        m_cCTrack1 = RawData.Track1;
        m_cCTrack2 = RawData.Track2;
        m_cCTrack3 = RawData.Track3;
        if(strstr(RawData.Track1,ParsedData.PAN))
        {
          m_cCTrackNumber = 1;
        }
        else
        if(strstr(RawData.Track2,ParsedData.PAN))
        {
          m_cCTrackNumber = 2;
        }
        else
        if(strstr(RawData.Track3,ParsedData.PAN))
        {
          m_cCTrackNumber = 3;
        }
        FireCardData();
        return (1);
      }
      else
      {
        fprintf(stderr,"<<CARD REJECTED>>\n\n\n");
      }
      EnablePayment();
      }
    }// of  if(pTNMsgHeader->pData && pTNMsgHeader->DataLength == sizeof(RawData))
  } // of   case TNMSG_CARD_DATA:
  break;
};
return((LRESULT)0);
```

```
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCCTrackNumberChanged()
{
        SetModifiedFlag();
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCCTrack1Changed()
{
        SetModifiedFlag();
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCCTrack2Changed()
{
        SetModifiedFlag();
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCCTrack3Changed()
{
        SetModifiedFlag();
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnPINRequiredChanged()
{
        SetModifiedFlag();
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCCFormatCodeChanged()
{
        SetModifiedFlag();
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCCPANChanged()
{
        SetModifiedFlag();
```

```
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCCNameChanged()
{
        SetModifiedFlag();
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCCExpDateChanged()
{
        SetModifiedFlag();
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCCServiceCodeChanged()
{
        SetModifiedFlag();
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCCAddlDataChanged()
{
        SetModifiedFlag();
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCCIssuerCodeChanged()
{
        SetModifiedFlag();
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCCIssuerNameChanged()
{
        SetModifiedFlag();
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::OnCardSuccessChanged()
{
        SetModifiedFlag();
```

```
}
////////////////////////////////////////////////////////////////////////
int CTNMagStripeCtrl::PinOK(char *_pinNumber)
{
  // This is is a function copied directly out of the TouchNet API
 // source library
  return (ParsedData.fPINOnCard) ? !tnStrCmp(ParsedData.AddlData,_pinNumber) : 1;
}
////////////////////////////////////////////////////////////////
short CTNMagStripeCtrl::InitControl()
{
  // in order for the control to receive messages from the
  // engine, it must open the Application Message Pipe
  pMsgUnit = tnMsgUnitCreate(TN_APP_MSG_PIPE_NAME);
  tnMsgUnitSetOwnerHWND(pMsgUnit,(HWND)GetHwnd());
        return 0;
}
////////////////////////////////////////////////////////////////
short CTNMagStripeCtrl::DeInitControl()
{
  // in order for any other application receive messages
  // from the engine, we must let the Pipe go.
    if(pMsgUnit)
  {
    tnMsgUnitDestroy(pMsgUnit);
  }
        return 0;
}
////////////////////////////////////////////////////////////////
short CTNMagStripeCtrl::EnablePayment()
{
  // This tells the Engine to turn on the entire Payment system
  // including the Card Reader, coin/cash reader and to take
  // only exact change  in the coin/cash reader
  int fsFlags = 0;
```

```
  fsFlags |= CASH_PAYMENT;
  fsFlags |= EXACT_CHANGE;
  fsFlags |= CARD_PAYMENT;
  return(tnapiCollectPayment(lNeeded-lBalance,fsFlags));
}
////////////////////////////////////////////////////////////
short CTNMagStripeCtrl::DisablePayment()
{
  //Tell the Engine to Stop accepting all forms of Payment.
  tnapiStopPayment();
       return 0;
}
////////////////////////////////////////////////////////////////////////////
short CTNMagStripeCtrl::UsePaymentCardTable()

{
  // Tell the Engine to use the Payment Card table to
  // validate cards
  return(tnapiUsePaymentCardTable())
}
////////////////////////////////////////////////////////////////////////////
short CTNMagStripeCtrl::UseIDCardTable()
{
  // Tell the Engine to use the ID Card table to
  // validate cards
  return(tnapiUseIDCardTable());
}
////////////////////////////////////////////////////////////////////////////
short CTNMagStripeCtrl::GetMajorVersion()
{
  // This will return the Minor Version Number
  return(MAJOR_VERSION);
}
////////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::SetMajorVersion(short nNewValue)
```

```
{
  // There would be no reason that the Web Space would need to Set a
  // new Major Version number
}
////////////////////////////////////////////////////////////////////////
short CTNMagStripeCtrl::GetMinorVersion()
{
  // This will return the Minor Version Number
  return(MINOR_VERSION);
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::SetMinorVersion(short nNewValue)
{
  // There would be no reason that the Web Space would need to Set a
  // new Minor Version number
}
////////////////////////////////////////////////////////////////////////
short CTNMagStripeCtrl::GetBuildNumber()
{
  // This will return the Build Number
  return(BUILD_NUMBER);
}
////////////////////////////////////////////////////////////////////////
void CTNMagStripeCtrl::SetBuildNumber(short nNewValue)
{
  // There would be no reason that the Web Space would need to Set a
  // new Build number
}
////////////////////////////////////////////////////////////////////////
short CTNMagStripeCtrl::EnableCardReader()
{
  // This will tell the engine to enable the Card Reader.
  int rc = tnapiCollectPayment(250,CARD_PAYMENT);
  return(rc);
}
```

Following is a print-out of representative source code at a Web site written to talk to the API according to one embodiment of the invention:

```
<html>
<head>
<title> Student Records Access Login </title>
</head>
<body topmargin="0" leftmargin="0">
<!-- This is the TNAPI Control for doing system wide things on the kiosk //-->
<OBJECT id="TNAPI" name="TNAPI"
classid="CLSID:C6DEBE13-59BC-11D0-A107-00A024545E65"
align="baseline" border="0" width="0" height="0">
</OBJECT>
<!-- This is the TNMAG Control for using the Payment system //-->
<OBJECT id="TNMAG" name="TNMAG"
classid="CLSID:5B4EE8B8-DE71-11D0-A6D2-006097D214C6"
align="baseline" border="0" width="0" height="0">
</OBJECT>
<script language="VBScript" for="window" event="onLoad()">
<!--
  'Hide windows that have 'TNHideMe!' as their window text
  TNAPI.HideWindows()
  'Turn on the payment functions
  TNMAG.EnablePayment()
  ' tell the System Manager to use the ID card table to validate the next card
  TNMAG.UseIDCardTable()
  ' initiaize the Card Control
  TNMAG.InitControl()
  'Don't allow the User to print this page
  TNAPI.DoNotPrint()
  ' tell the engine to log and 'Application Begin' for this application
  TNAPI.ApplicationName="SRLOGIN"

  ' Prompt the user to put in a card
  CardStatus.CCStatusLine.value = "Please Insert Card"
//-->
```

```
</script>
<script language="JavaScript" for="window" event="onUnload()">
<!--
  // Let the Card control release its resources
  TNMAG.DeInitControl()
//-->
<script language="JavaScript" for="TNMAG" event="CardInserted()">
<!--
  // A card has been inserted in to the card reade,
  // instruct the user to remove the card.
  CardStatus.CCStatusLine.value = "Remove Card Quickly!"
//-->
</script>
<script language="JavaScript" for="TNMAG" event="CardError()">
<!--
  // A Card error has occurred, tell the user.
  CardStatus.CCStatusLine.value = "Card Error"
//-->
</script>
<script language="JavaScript" for="TNMAG" event="CardInvalid()">
<!--
  // The card is invalid, tel the user
  CardStatus.CCStatusLine.value = "Card Invalid!"
//-->
</script>
<script language="JavaScript" for="TNMAG" event="CardDisable()">
<!--
  // The Card Reader has been disabled, tell the user.
  CardStatus.CCStatusLine.value = "Card Reader Disabled!"
//-->
</script>
<script language="JavaScript" for="TNMAG" event="CardEnable()">
<!--
  // The Card Reader has been Enabled, tell the user.
  CardStatus.CCStatusLine.value = "Card Reader Enabled"
```

```
//-->
</script>
<script language="JavaScript" for="TNMAG" event="CardReset()">
<!--
  // The Card Reader has been Reset, tell the user it is ready.
  CardStatus.CCStatusLine.value = "Card Reader Enabled"
//-->
</script>
<script language="JavaScript" for="TNMAG" event="CardTest()">
<!--
  // The Card Reader is being tested, tell the user.
  CardStatus.CCStatusLine.value = "Card Reader Enabled"
//-->
</script>
<script language="JavaScript" for="TNMAG" event="CardData()">
<!--
  // The Engine has validated a card
  CardStatus.CCStatusLine.value = "Please Wait . . ."
   // take the Personal Account Number (PAN) from the Card Control
  // and put it into the Form
  Form.ID.value = TNMAG.CCPan
 // Submit the form
  Form.submit()
//-->
</script>
<form action="tsrvweb.exe" method="POST" name="Form">
<input type="hidden" name="tserve_tip_write" value="||ID">
<input type="hidden" name="tserve_trans_config" value="k_menu.cfg">
<input type="hidden" name="tserve_response_template" value="k_menu4.htm">
<input type="hidden" name="tserve_host_code" value="0">
<input type="hidden" name="ID">
<input type="hidden" value="  Reset  ">
        <!-- Submit Button -->
        <p> <input type="submit" value="TNHideMe!"> </p>
        </form>
```

```
<!-- This is simply a form to give feed back to the user //-->
<form method="POST" name="CardStatus">
<p> <input type="text" size="30"
        name="CCStatusLine"
        value="Insert Card Now"> </p>
</form>
</body>
</html>
```

Although these descriptions have focused on use of the system of the present invention as a tool for permitting access by and control of local public access terminals or kiosks by Web sites, It should be appreciated that the system of the present invention is capable of being incorporated in the form of a variety of embodiments, only a few of which have been illustrated and described above. The invention may be embodied in other forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive and the scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

CLAIMS:

What is claimed is:

1. A method for adding the functionality of a local accessing terminal to web browser based applications on a Web site, said method comprising the steps of:

creating an applications program interface (API) using standard Web technology protocol to act as a message carrier between the Web site and the functionality of the local accessing terminal;

installing said API such that it can communicate with the Web site through a browser;

establishing a connection between the local accessing terminal and the Web site thereby triggering said API; and

using the API to communicate with the Web site the availability of functionality on the local accessing terminal.

2. The method of Claim 1 wherein the API resides on the local accessing terminal.

3. The method of Claim 2 comprising the further step of the API signaling the Web site of the availability of local accessing terminal functionality by sending data, code, or one or more objects to the Web site through the browser.

4. The method of Claim 2 comprising the further step of the Web site, uponinterconnection with the local accessing terminal, using the API to poll the local accessing terminal through the browser to search for signatures indicating local accessing terminal functionality.

5. The method of Claim 1 wherein the API resides on the Web site.

6. The method of Claim 5 comprising the further step of the Web site, upon interconnection between the Web site and the local accessing terminal, sending the API to the local accessing terminal.

7. The method of Claim 5 comprising the further step of the Web site using the API to search for signatures at the local accessing terminal.

8. The method of Claim 1 comprising the further steps of:

the Web site providing Web page screens to the user that are appropriate for the functionality available at the local accessing terminal;

the Web site providing web page screens to the user that are responsive to the user's input; and

the Web site optionally activating and controlling functionality at the local accessing terminal.

9. The method of Claim 1 comprising the further steps of:

the API communicating to the Web site that the local accessing terminal has only standard personal computer functionality;

the Web site providing web page screens to the user that are appropriate for the functionality available at the local accessing terminal; and

the Web site providing web page screens to the user that are responsive to the user's input.

10. The method of Claim 1 comprising the further steps of:

the API communicating to the Web site that the local accessing terminal is a functionally rich local access terminal;

the Web site providing web page screens to the user that are appropriate for the functionality available at the local accessing terminal;

the Web site providing web page screens to the user that are responsive to the user's input; and

the Web site optionally enabling, disabling or controlling said functionality in response to the user's input.

11. The method of Claim 1 wherein the API is written in ActiveX.

12. The method of Claim 1 wherein the hardware and software of the local accessing terminal are managed by local system manager software.

13. The method of Claim 12 comprising the further steps of:

the API communicating with the local systems manager software;

the local systems manager software communicating with the softwar and hardware of the local accessing terminal.

14. A system for enabling a Web site to communicate with and optionally activate and control hardware and software located on a local access terminal without having to use a particular browser or browser plug-ins, said system comprising an applications program interface (API) in communication with the Web site through a browser and with the hardware and software of the local accessing terminal.

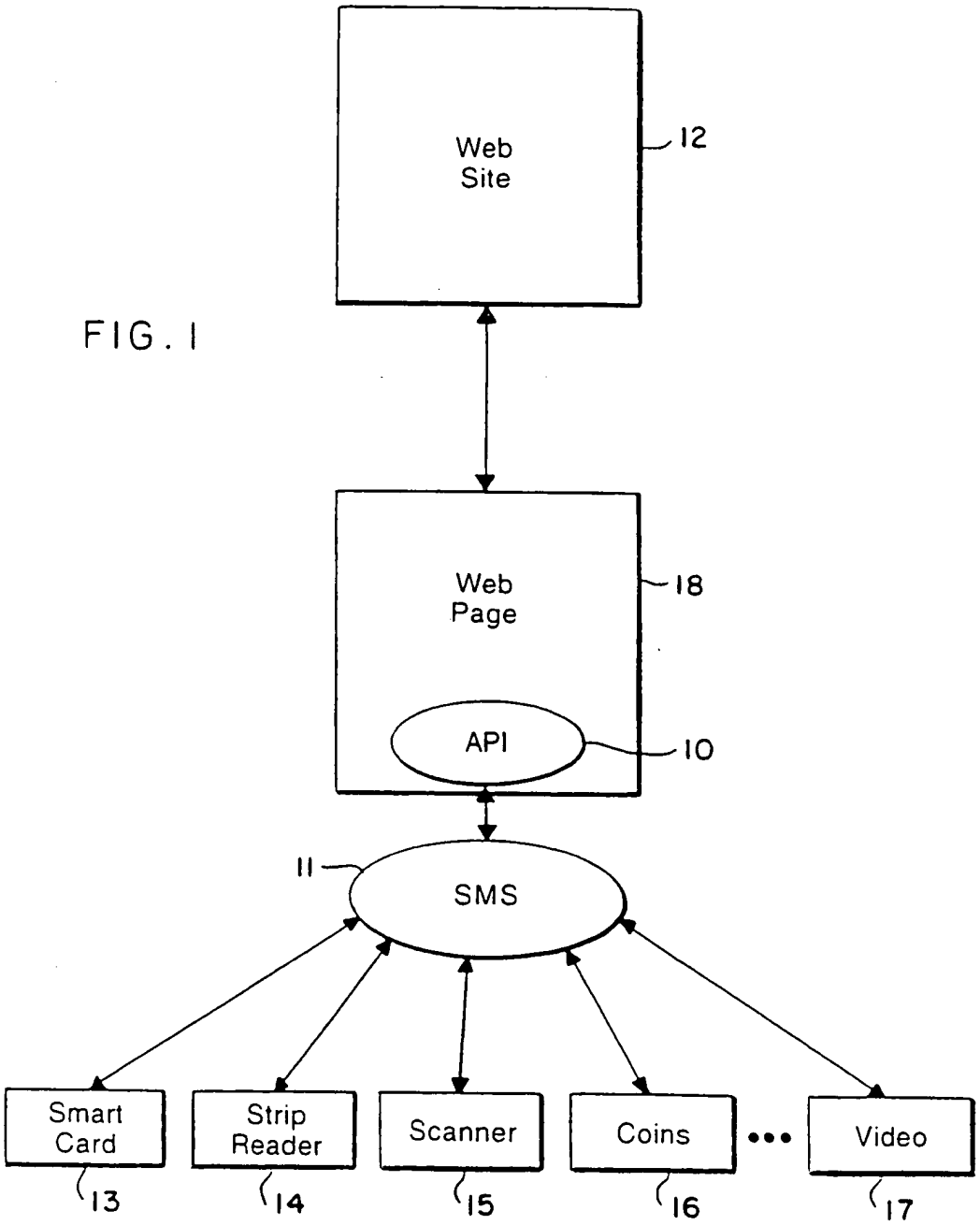
15. A system for enabling a Web site to communicate with and optionally activate and control hardware and software located on a local access terminal, said system comprising:

at least one local accessing terminal which utilizes a browser for communication with said Web site;

an applications program interface (API) using standard Web technology protocol accessible to the local access terminal; and

a Web site providing the intelligence to exercise control over the local access terminal functionality.

FIG. 1



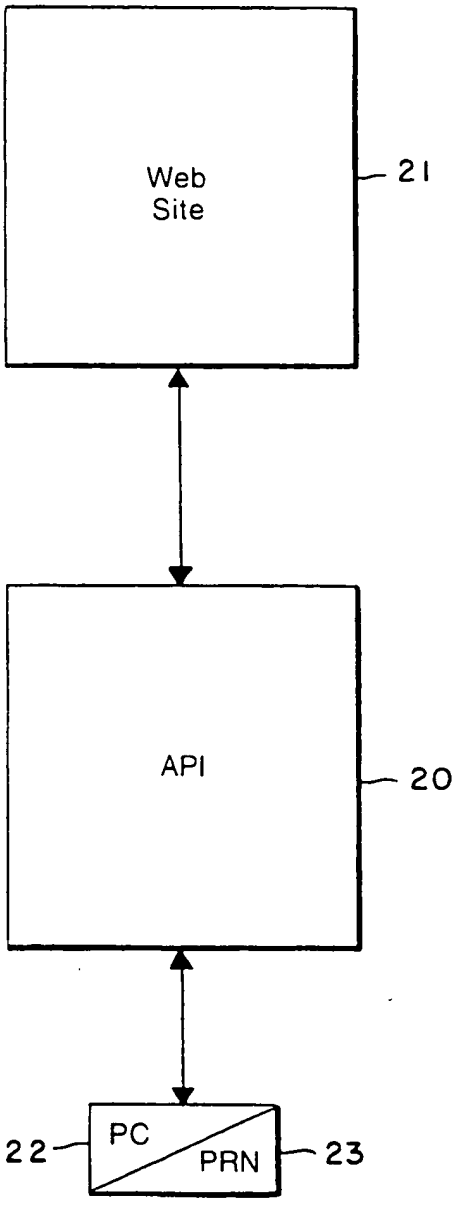


FIG.2

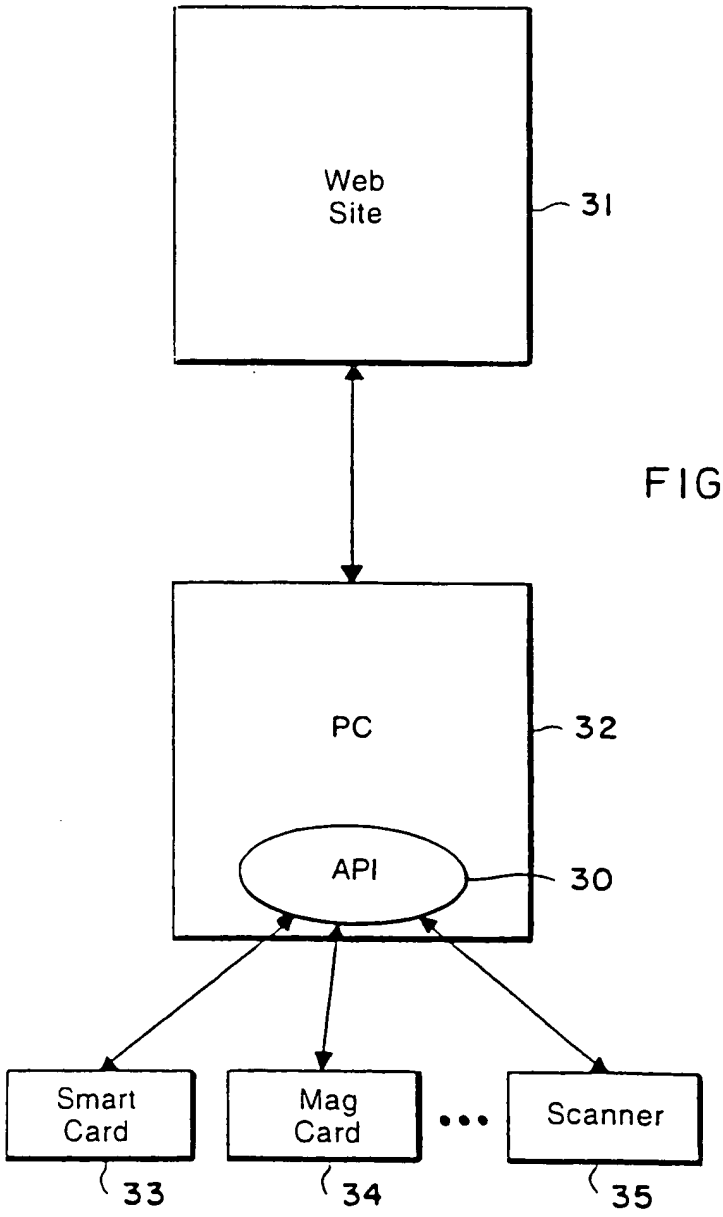


FIG. 3

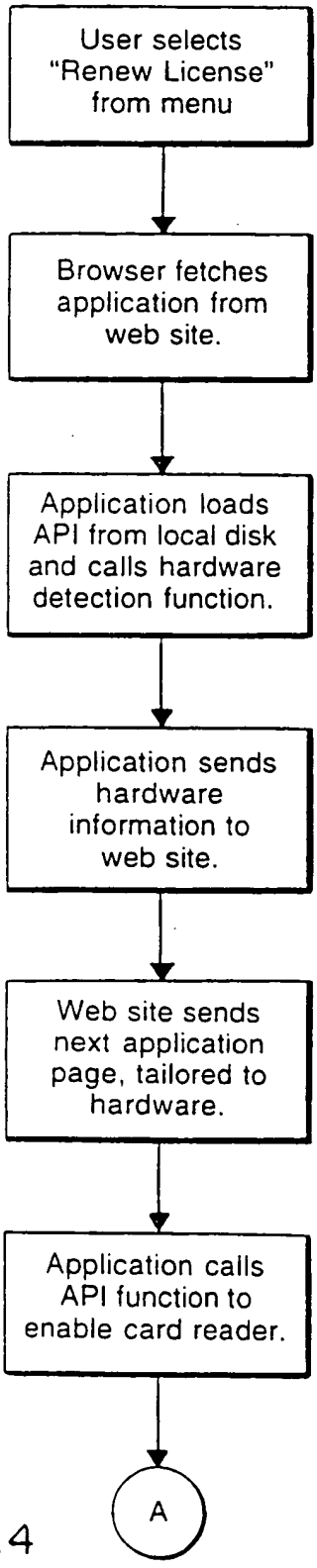


FIG.4

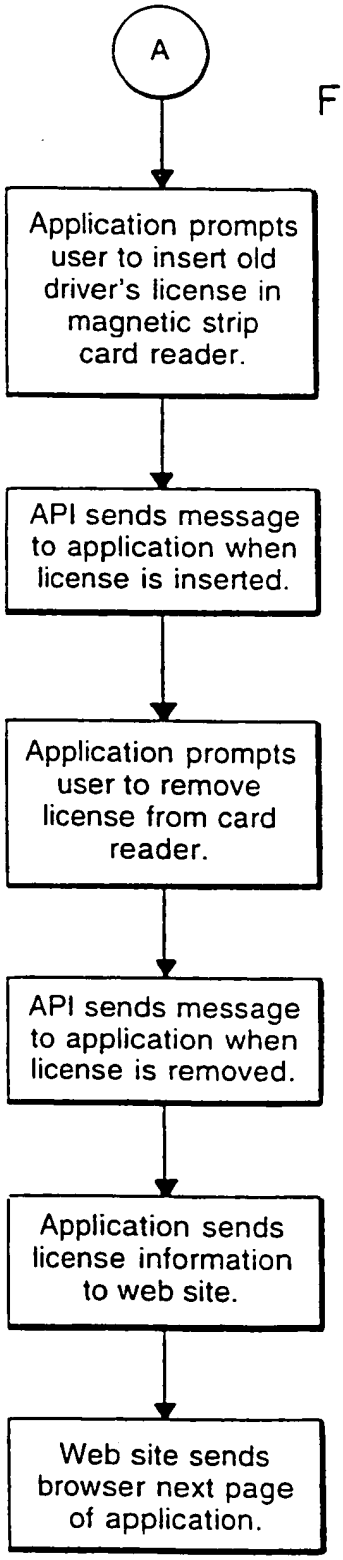


FIG.5

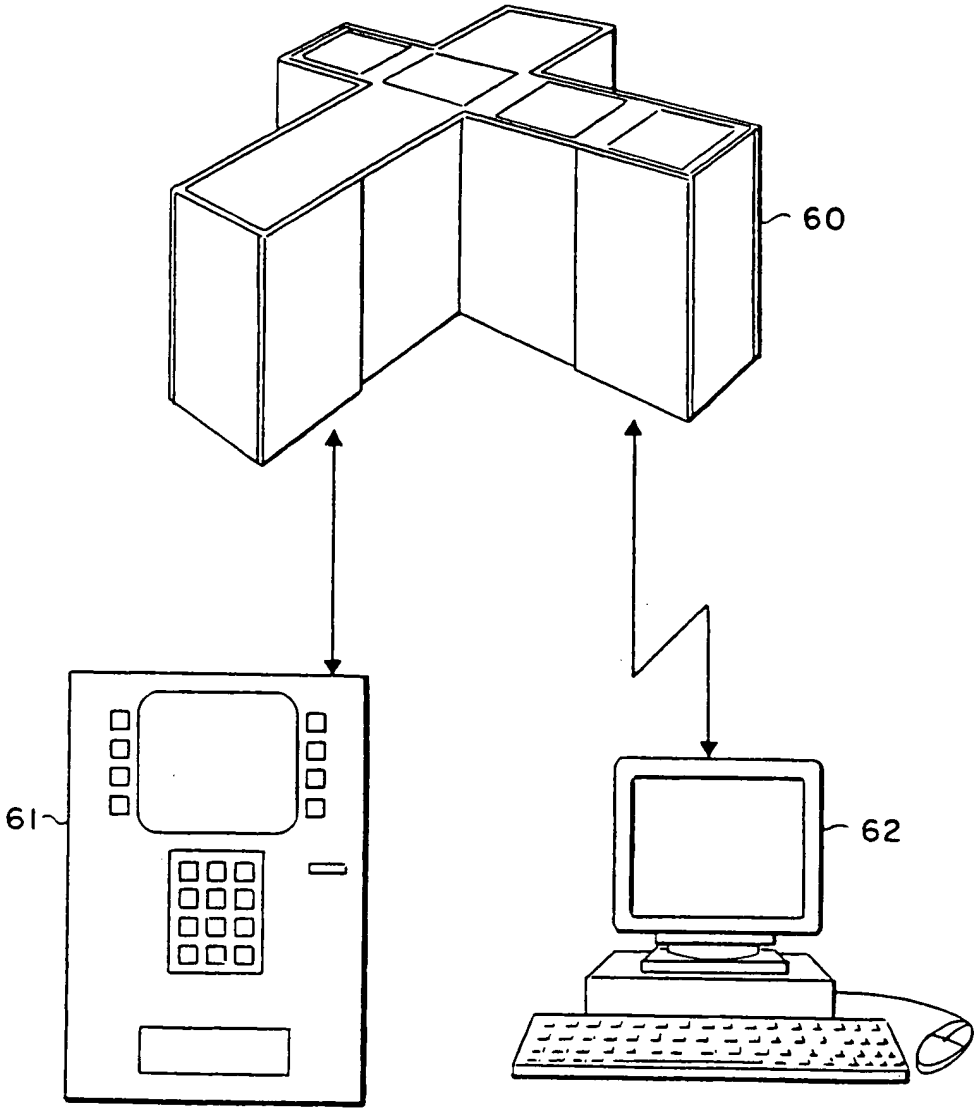


FIG. 6

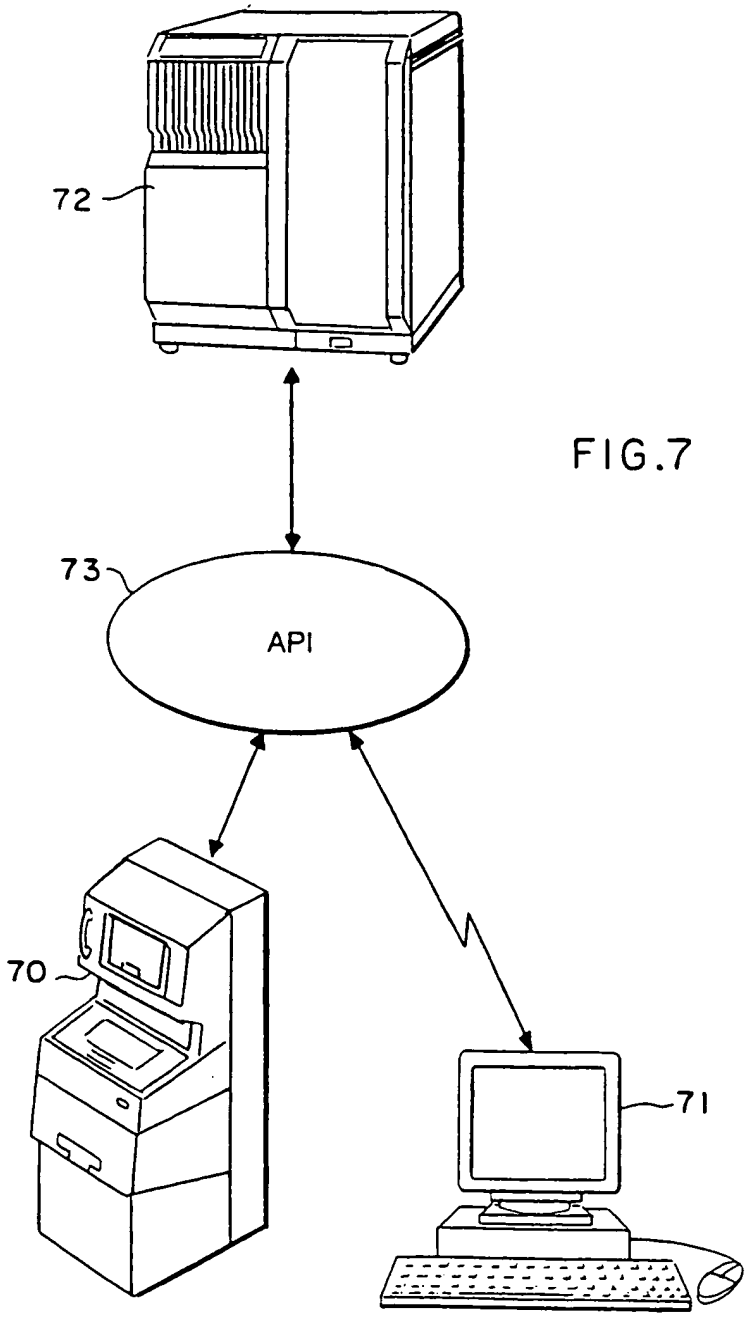


FIG.7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/19611

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 13/00
US CL :395/682, 200.48; 345/335
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/682, 200.48; 345/335

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,430,836 A (WOLF ET AL) 04 JULY 1995, Abstract, col. 3, lines 3-12, col. 4, line 66 - col. 5, line 21.	1-15
Y	US 5,572,643 A (JUDSON) 05 NOVEMBER 1996, col. 3, lines 44-57, col. 4, lines 36-51.	1-15

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 02 DECEMBER 1998	Date of mailing of the international search report 20 JAN 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer LE HIEN LUU Telephone No. (703) 305-9650

Form PCT/ISA/210 (second sheet)(July 1992)*